NASA-TM-112714

JOURNAL OF COMPUTER & SOFTWARE ENGINEERING, 1(4), 325–348 (1993)

# Reliability Analysis of the X-29A Flight Control System Software

**Gloria J. Davis**

NASA Ames Research Center
Moffett Field, CA

**Michael R. Earls**

NASA Dryden Flight Research Facility
Edwards Air Force Base, CA

**F.A. Patterson-Hine**

NASA Ames Research Center
Moffett Field, CA

Software reliability measurements of safety-critical software systems are not well understood. In particular, a significant part of the testing of flight control software for high performance aircraft is performed in full-up systems tests, so the applicability of models developed for pure software systems is unknown. In this study, data from flight tests of the X-29A forward-swept wing aircraft, performed at NASA's Dryden Flight Research Facility, are analyzed with the Statistical Modeling and Estimation of Reliability Functions for Software (SMERFS) modeling package. This initial study is performed to identify the applicability of current statistical software reliability models to this type of data. Results from this analysis are presented following a description of the data collection and documentation process utilized by the X-29 program. These results are used to assess the applicability of these models and their prediction capabilities in a flight test environment. Conclusions are presented, comparing the use of the statistical reliability models with the existing operational technique of testing with analysis using an engineering perspective.

reliability analysis, safety-critical software, testing of flight control software, statistical software models

## 1 INTRODUCTION

NASA Ames Research Center conducts research on techniques for assessing the dependability of safety-critical systems. To date, the tools and techniques have been applied primarily to hardware systems. With software increasingly used as a critical component of safety-critical systems, reliability analysis of software in an integrated system context is necessary. Much of the work in software reliability assessment reported in the literature has been done with a focus on commercial systems. To determine the applicability of such models to safety-critical systems, the X-29A flight control system was selected for analysis.

The flight operations community at Ames' Dryden Flight Research Facility conducts flight research tests on experimental aircraft. Their successful process of ensuring safety throughout the research programs has evolved through the years and is a culmination of experience and good engineering judgment. Their subjective analysis approach to achieving qualification of each platform works, as it has never caused nor failed to detect problems that resulted in catastrophe. However, the testing process for such systems at Dryden employ no objective means of measurement and hence no metric to aid in decisions to continue or to stop testing. Their current techniques for system verification and validation, as particularly noted for safety-critical software, do not consist of strict formal analysis techniques.

This paper presents the current safety-critical software evaluation techniques employed at Dryden, complete with their development and test philosophy. Then an analysis technique of "engineering judgment" is applied to the currently maintained data and presented in the form of "quick look" graphs for further analysis. Next, formal statistical reliability analysis models are applied to these same data to determine the usefulness of their predictive nature to this community, and to determine if this new representation of the data could or would affect the flight test programs. The results show that the information provided by these methods could help the program if the models could provide predictions earlier in the software test process. The information provided by these models would enable the test engineers in their efforts not necessarily in determining when to fly, but when *not* to fly. Future work will determine what other types of programmatic information could be used by the models to enable more accurate results earlier in the test program.

The following section provides an overview of the X-29 program. Next is presented the software testing strategy followed by the failure reporting process. The data obtained during the testing process are then segregated and analyzed with engineering judgment using quick look data graphs, with the results reported in Section 5. Section 6 presents the results of executing representative models in the SMERFS software package with this data. General observations from this analysis are then summarized in the concluding section.

## 2 X-29 PROGRAM OVERVIEW

NASA Dryden Flight Research Facility (DFRF) specializes in the flight test research of high performance aircraft. The X-29A forward-swept wing technology demonstrator, an aerodynamically unstable aircraft requiring a highly augmented software-intensive control

system to maintain stable flight, was initially flight qualified and operated from 1984 to 1988. The X-29 was developed under a DARPA sponsored program and managed by the U.S. Air Force Flight Dynamics Laboratory. The system was contracted to Grumman Aerospace Corp., who, in turn, subcontracted the sensor computer subsystem to Honeywell, Inc. Honeywell was responsible for the flight control computers, flight software, and control system sensors. DFRF conducted reviews and the flight test program.

Designed to be fault tolerant, the flight control system was a TMR (triple modular redundant) digital system with a triplex analog backup system. Each of the three digital flight control computers contained two digital processors—the control law processor and the input/output processor. The two digital processors were Honeywell HDP-5301 processors and the software for them was written in HDP-5301 assembly language. The operational flight program contained approximately 29,000 instructions. The system was designed to be fail operational/fail safe, hence able to return safely to base after any two failures.

## 3 SOFTWARE TESTING STRATEGY

Data collection by DFRF on the X-29A flight control system began with the initial flight qualification and continued through operational maintenance phases. For flight qualification, there are two types of testing the flight software must pass: verification and validation (V&V). Recognized as a critical system component, the flight control software was subjected to the contractor's V&V as well as NASA's independent V&V. Verification ensures that the software operates as specified, through module testing, internal independent review, and systems testing. The validation process ensures that the delivered system performs adequately to accomplish the flight requirements. This includes failure modes and effects tests in which failures are artificially induced and observations are made as to the proper system response. Recognizing that the possible known combinations of real-world scenarios are too numerous to analyze completely, continuous system testing in a simulated environment was key to assuring that the generic faults would more likely be discovered on the ground than in actual flight tests. Validation tests concentrated on the most critical elements of the software: the control laws during normal operation, fault reaction, mode logic, and redundancy management of computers and sensors.

NASA's primary V&V tool was the real-time six-degree-of-freedom hardware-in-the-loop simulation facility. With three flight control computers (two flight qualified), a failure status control panel, and an analog computer model of surface actuators, the full range of flight phases were executed. This independent facility was used to identify most of the discrepancies used in this analysis of software reliability. The simulators were run most every day, with a log kept as to the number of hours the flight computers were running. The data presented are from both the simulation laboratory and the actual flight schedule. No data from the development phases of the project are included in this analysis.

The process and techniques used in the X-29 project to assure safe, reliable software are found in detail in a NASA Technical Memorandum, generated by the software manager at Dryden [1]. There was no formal analysis on the reliability of the flight control software. As indicated by Duke [2], confidence is gained incrementally that the system

will function correctly and reliably through Dryden's established qualification, verification and validation process. The software of the X-29 was developed to the written specifications and tested via a software test matrix. System tests were also generated from the same specifications and executed on the flight control system software to ensure that all of the requirements were implemented as specified. The data used for this analysis were from the full-up system test.

### 3.1 Flight Control Software and the Flight Schedule

In the flight qualification and operational test phases at Dryden, the flight control software was released in versions, building upon the previous version by increasing the flight envelope, enhancing the type of research capabilities, and repairing detected errors. The software was strictly maintained and the modification history for each module was recorded in the code. More than 100 software changes were implemented from the first submission. All flown software changes were made by the subcontractors who designed and developed the code.

The flight schedule was designed to test and evaluate the aircraft's performance in an expanding operational envelope. There were 14 versions of software flown over the course of 242 flights. The first flights were of minimal functionality with a limited flight control window and flying 0.6 Mach at 30,000 feet, to basically prove the safety of the craft. After incrementally establishing the basic safety of the craft, the first full flight envelope was executed after flight number 16, with the third flight software release version. A complete list of the software versions, number of flights and the test objectives is provided in Table 1.

## 4 ERROR DETECTION/DISCREPANCY REPORTS

A discrepancy reporting system was established to identify occurrences that were either not expected or considered unsafe. When an unexpected result or problem occurred during testing or flight, engineers would describe the problem, indicate which configuration item was the probable cause (of 20), assign a criticality to the problem, and date the discrepancy report (DR). Each DR was recorded in a database and submitted to a configuration control board. After corrective action was accomplished and successively tested, it was resubmitted to the board to be closed out. There were three levels of criticality that discrepancies could have: criticality 1 is flight critical, indicating that a loss of life or aircraft could occur; criticality 2 is mission critical, indicating that the research objective could be compromised; and criticality 3 is any other detraction from the specifications that is not life or mission critical, but rather a relative nuisance.

### 4.1 Discrepancy Classification

In the flight qualification and operational testing phase of the X-29 at Dryden, there were 296 discrepancy reports generated covering 20 different configuration items (i.e., navigation, instrumentation, flight controls system, structure, etc.), the first one was dated

**Table 1. Flight Software Versions and Flight Schedule**

| Flight Schedule SW Version | Number of Flights | Flight Time (hours) | Altitude Range (ft) | Max. Velocity (Mach) | Flight Test Objective |
|---|---|---|---|---|---|
| III-AI-00 | 8 | 11.2 | 15K–20.3K | .43–.62 | functional test and handling qualities |
| III-AJ-01 | 8 | 10.2 | 16K–31K | .58–.62 | ASE & FCS Stab. Clearance, Formation, Flow Visualization |
| IV-AC-03 | 7 | 8.7 | 30K–41K | .70–.83 | Flight envelope exp. |
| V-AB-00 | 1 | 1.5 | 40K | 0.9 | man. camber contr. |
| V-AB-01 | 13 | 13.7 | 30K–41K | .73–1.1 | envelope exp. |
| V-AC-01 | 2 | 2.3 | 40K | 1.05–1.1 | sw verif., env. exp. |
| VI-AA-00 | 32 | 23.6 | 15K–40K | .85–1.4 | envelope expans., maneuver expans. |
| VII-AC-00 | 16 | 9.3 | 8K–43K | .9–1.2 | envelope expans., diver. envel. clear. |
| VII-AD-00 | 17 | 13 | 15.5K–50.2K | .7–1.46 | handling quals., lds. expn., Navy eval. |
| VII-AD-01 | 38 | 25 | 12.6K–50.2K | .92–1.39 | aoa, airspeed, perf. |
| VIII-AB-01 | 44 | 29.7 | 17.8K–50.4K | .91–1.48 | loads expans., engine vibration |
| VIII-AC-04 | 10 | 5.2 | 24.8K–46.2K | 1.05–1.31 | SW test, MCC div. |
| VIII-AD-00 | 17 | 12 | 15.4K–46.2K | .96–1.2 | wing/canard loads |
| VIII-AF-00 | 29 | 21.9 | 16.4K–41.7K | .6–1.2 | loads clear/milit. util. |

06/20/84 and the last 1/27/89. The first flight was successfully completed on 12/14/84. At the time of first flight, the software had already been tested by the contractors and by the Dryden program engineers, and a calculated 380 hours of full-up systems testing had been logged.

It should be noted that not all of the reported DRs were failures. Each observed discrepancy was perceived by test engineers to have a chance of causing failures. Upon review, most of the discrepancies were closed by corrective action or with a decision that no action was required. Others were either eliminated because the flight mode in which they occurred was removed or were never closed, remaining open because they could not be resolved. The latter were typically carried on each flight as a Hazard Report (HR) and acknowledged as an accepted risk. An example of a HR is "if the dual pitch rate gyros fail to null and the 2 failures are undetected without commanded motion then this could cause loss of control." This was determined as having a low probability of occurring in a 1-hour flight when the pilot is not generating much pitch rate (2.40/s). The causes of the DRs ranged from tolerance levels set too conservatively resulting in premature shutdown or false failures, to wrong implementation of a test (not a system failure but a test failure).

An important factor also to be noted in this analysis is the evolutionary aspect of the software. The X-29A was designed to be a research tool. Assuring that the tool was safe was of primary importance and once established, the research on the tool was performed in an evolutionary manner. The flight control system software performed critical functions in both the safety and research aspects of this tool. Hence the software was constantly evaluated for safety and enhanced to enable further research. While the operational safety

was never compromised, the research envelope was continually pushed. Hence there were 58 to 60 versions of software that were released (qualified as available for flight) and 14 actually flown.

As with any component of the aircraft, one must expect downtime to repair the software. It was determined not unreasonable to expect 25% downtime to software.[1] This is logical because of the repair cycle involved. Once a problem is detected, the data are first retrieved from the flight computers. After the data are analyzed and the source of the problem identified, the appropriate changes in the software are implemented. After the software is changed, the modifications are subjected to the appropriate V&V tests. Once they satisfactorily pass, the entire flight control system is retested in the full-up systems test. Only after successfully passing this repair and test process is the software reinstalled into the aircraft and flown again.

## 5 RAW DATA ANALYSIS

The data presented in this report were not collected and maintained by the X-29 program managers and engineers for purposes of statistical reliability analysis. There was no formal reliability analysis identified as performed within the context of the program at DFRF, separate from the full-up systems testing that has to date enabled program success. These data were assembled at NASA Ames Research Center through an iterative process of explaining our interests (software reliability analysis), indicating the various formats for the data that would be useful, and clarification by the project team about their information collection process. This effort was an assembly of various pieces, all of which were readily available (though by different sources and different places). The data were found through calls to key project personnel, access to the full set of discrepancy reports, the flight schedule, software release versions, and the system simulation log. The data span 6/20/84, by which time 102 test hours had elapsed, to 1/27/89 with 2863 elapsed CPU hours. This last date was that of the last flight control software discrepancy for the X-29A ship 1.

The first analysis is performed in a very subjective fashion: engineers (software, aeronautical, and mechanical) looking at the data, understanding the functionality of the system, and justifying the occurrence of various discrepancies. The graphs of the raw data are presented without mathematical manipulation or curve fitting applications. The conclusions and abstractions made from this use engineering judgment and insight.

### 5.1 Discrepancy Reports Breakdown

As previously stated, 296 discrepancy reports were recorded at Dryden. The breakdown by percentages of these is given in Figure 1, indicating that 30% of the total were due to the flight control system, 65% of which were software related, and of these 21% were safety critical.

Further breakdown of the discrepancies by configuration item, shown in Figure 2,

[1] Ed Collins, X-29A AJPO and program historian, Wright Patterson AFB, personal communication, Oct. 1991.

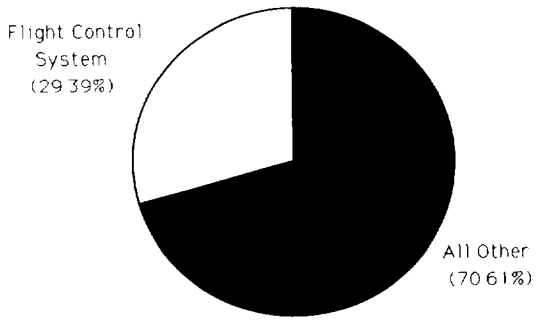


**Figure 1a. X-29A discrepancy reports.**

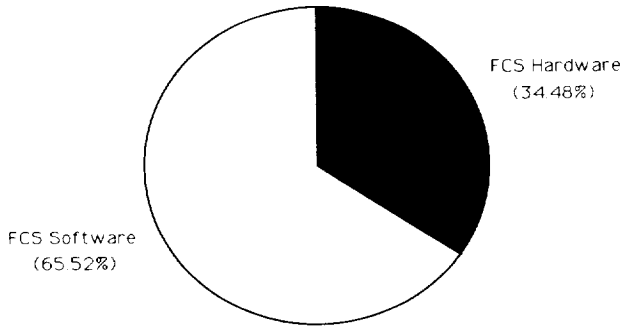


**Figure 1b. Flight control system DRs.**

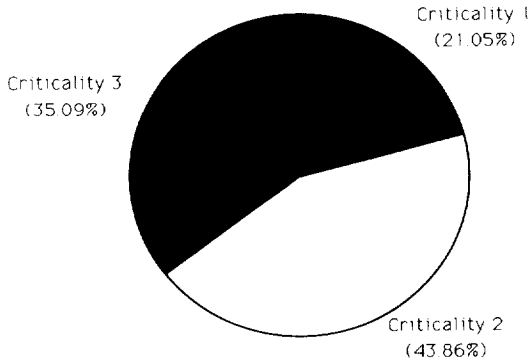


**Figure 1c. Software DRs.**

indicates that the greatest percentage was due to instrumentation. However, a closer look shows that these were all of criticality level 2 or 3. Sixty percent of the criticality level 1 discrepancies were due to the flight control system software. The amount of discrepancies for software alone indicates at least two major points: that the full-up systems simulation is critical to the discovery of software errors that may only be discovered in such a setting, and that the incorporation of reliability assessment measures at this phase of the program could be beneficial. The ability to predict the occurrence of the next failure or discrepancy

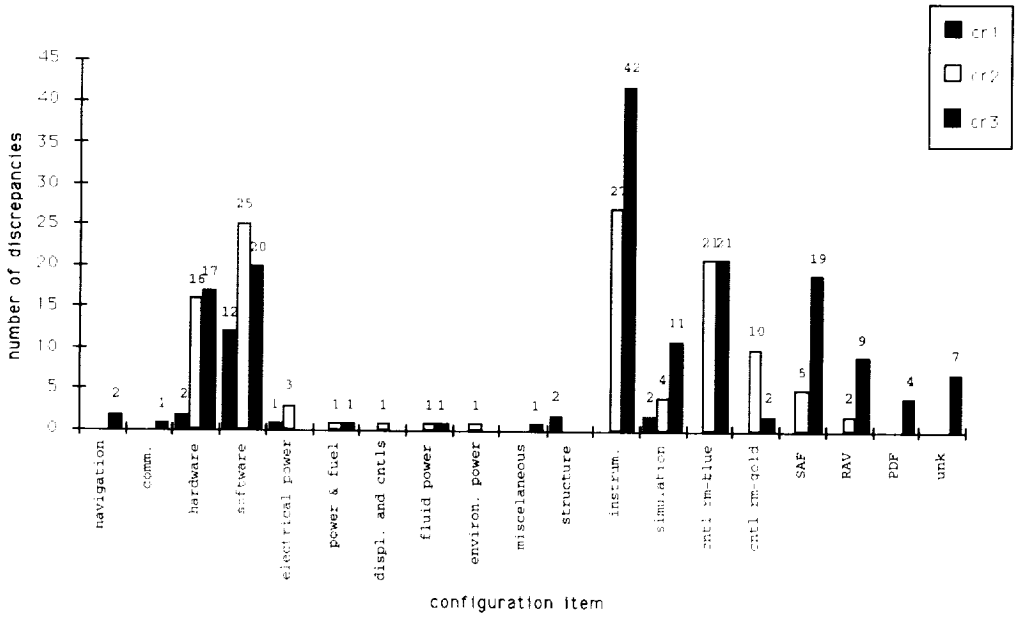


**Figure 2. X-29A aircraft systems discrepancy reports.**

could provide program engineers insight to problem areas during their test efforts, and enable more focused testing on these areas.

### 5.2 Flight Control Software Discrepancies

The Dryden community is acutely aware of the importance of data collection. However, the collection and storage of this information for subsequent traceability is not the end of its usefulness. The graphic representation of such data provides an intuitive overview of the trend in the establishment of reliability in a system. If the cumulative number of discrepancies is increasing at a constant rate, then the overall system maturity is not being established. However, an indication that the system is gaining in reliability is that the slope of the overall curve tends to decrease in magnitude, levels off to some asymptote, or ideally, remains constant at 0. Looking at the slope of the cumulative discrepancies is not the only indication of achieved reliability. It must also be understood what is being performed in the course of generating this curve. In the case of Dryden's test program, all of the information presented in the following set of graphs is from the established full-up systems testing of a research aircraft system in which complete or subsets of actual flight profiles are conducted.

### 5.3 FCS Criticality Levels and Detection Dates

As seen in the graph of their actual time of detection, shown in Figure 3, the overall error detection rate is constant, but the rates of detection for the individual criticality levels are not constant. By the time of the first test flight (12/14/84, simulation execution hours 380), 41% of the total number of detected criticality 1 errors had been found, compared

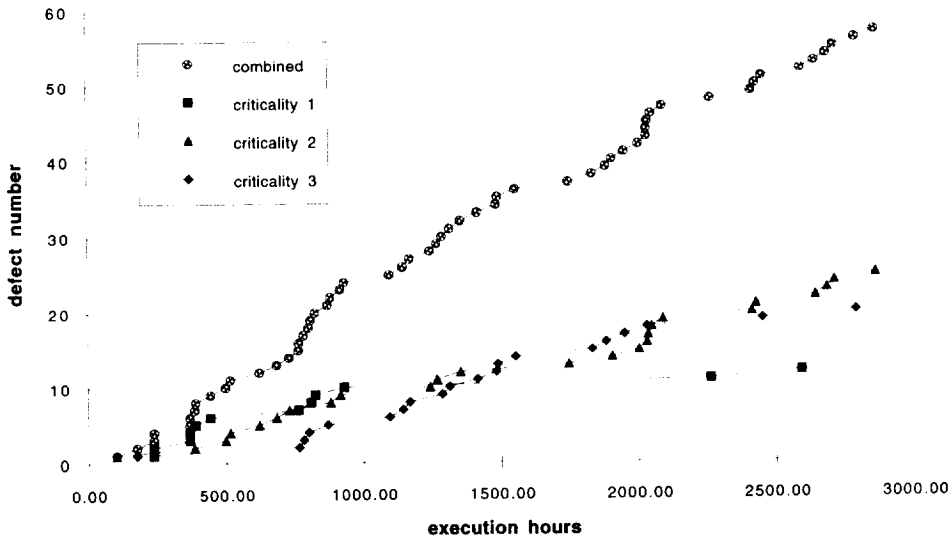


**Figure 3. Cumulative software DRs vs. execution time.**

with 8% of the criticality 2 and 5% of criticality 3 DRs. Over 80% of the criticality 1 DRs were submitted in the first 40% of collection time, whereas 85% of the reported criticality 3 DRs were in the middle 40% time span. The criticality 2 discrepancies tended to be detected at a more constant rate overall. It seems intuitive that for safety-critical systems, the visibility of certain errors is not realized until a certain amount of more critical errors are discovered. This behavior has been stated before and classified by testing technique [3]. This is not true for all errors, however, because the last two criticality 1 errors were probably present from the beginning of the flight program, but they were neither enabled nor discovered by the to-date scenarios flown or simulated. An example is the last criticality 1 DR which was a single point of failure discovered after over 3½ years of flying the craft (calendar time). The problem was described as a power supply detection logic problem which would have resulted in loss of the control of the aircraft had a single failure to null of the digital impact pressure input occurred. It was discovered in V&V testing of a new flight release software version yet had been resident in the system from the first release.

### 5.4 Seasonal Trend Detection

Over the 56 months the data were collected, the graph shown in Figure 4 indicates the monthly density of the software discrepancies for all levels of criticality, yielding an average of one discrepancy per month. This plot of data by the month could readily show seasonal trends within the data. As can be seen, no trend strictly based on seasons are obvious to this data set.

### 5.5 Failure Intensity

Another graphical representation of the raw data that provides immediate trends toward increased reliability is the total number of DRs divided by the elapsed time to date. The

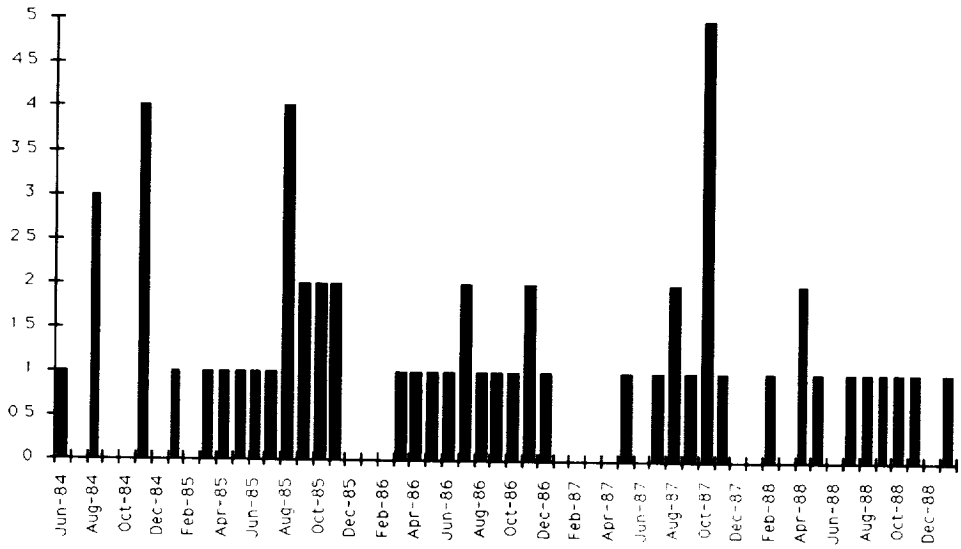


**Figure 4. Monthly density of software defects.**

plot of this failure intensity ($\lambda$) to execution time for the DRs further adds credence to the idea that the more critical errors are discovered early in a testing program, with less critical errors observed only after the more critical ones are uncovered. The visually constant rate of the failure intensity for all levels of criticality is shown in the top curve in Figure 5. The fluctuations are consistent with the V&V simulator tests on the new versions of the software, whereby additional software was incorporated and/or an expanded research flight envelope was being tested. A closer look at the individual criticality levels is important for this program. Remember that this aircraft is a research tool. However, to enable research it must first be safe. Therefore, the criticality levels indicate the order of importance, that is (1) safety, (2) research, and (3) other.

Figure 5 presents a graph of the failure intensity of the total set of DRs and each of the criticality 1, 2, and 3 subsets of DRs. The criticality 1 DRs have a rapidly decreasing rate with time. Maintaining safety is the primary goal of a safety-critical system. Hence a rapid decrease in criticality 1 DRs essential. The intensity of the criticality 2 DRs indicates a more constant rate with a tendency at the end towards an increase. With the continued expansion of the research capability, testing new scenarios or adding further capability explains the increase in criticality 2 DRs. The graph of the criticality 3 DRs shows a delayed start, with a slow peak, and then a tendency toward decrease. It was indicated by the program office that as time passed, tradeoffs between reporting nuisance errors with formal traces of them and tolerating them were weighed and the latter increasingly would be chosen without compromising the safety or research capability of the aircraft. Looking at the individual failure intensity curves it is seen that this is due to the discovery of the criticality level 2 DRs. The intensity of the safety critical DRs provided a classic software reliability growth, highlighted in Figure 6. Based on the cumulative curves in Figure 3, these trends are all understandable in the context of this experimental flight test program.

Each of these graphs make sense in the context of the program because of the implications of failures and the program's testing philosophy. Although this aircraft is a research platform, safety is of first and foremost importance. The flight control system was

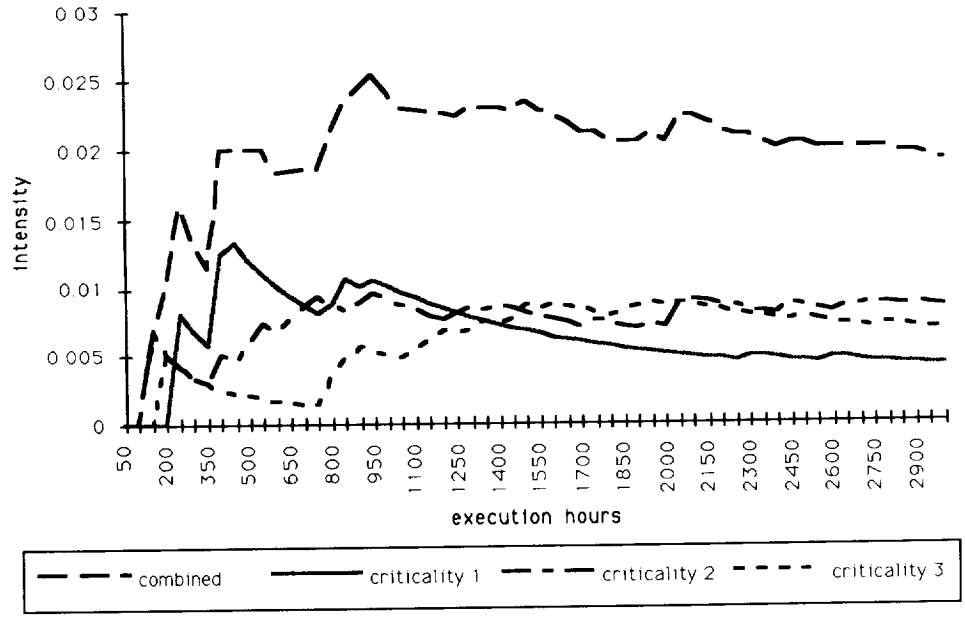


**Figure 5. Failure intensity by criticality level.**

required to successfully complete an established test matrix after system change. Test sets were continually updated to reflect and test newly found errors. This enabled the program engineers to gain confidence in the safety of the aircraft. Subsequent to this, the research envelope was continually expanded through updated versions of the software. The constant growth of capabilities lends itself to a constant level of mission critical discrepan-

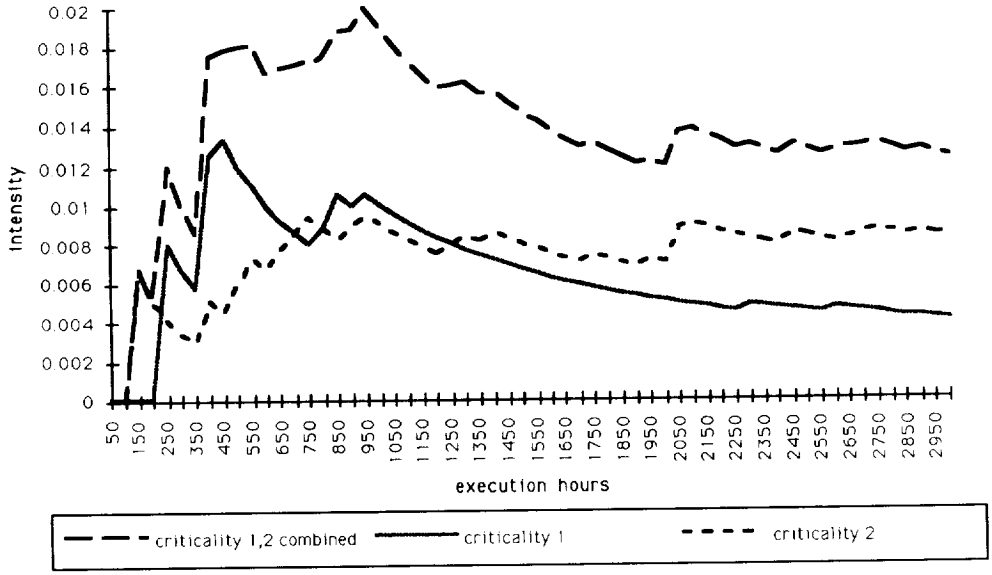


**Figure 6. Failure intensity of criticality levels 1 and 2.**

cies. The criticality level 3, or relative nuisance errors, tended to be identified after the safety comfort level was established and detection continued semiconstantly. Admittedly, more of this type could have occurred yet not been reported, accepting that some nuisance errors are more tolerable than worth the effort of officially reporting with follow-up repair and closure.

Discrepancies of all criticality levels contain useful information; however, to gain insight into flight critical systems, the lack of implication by the nuisance discrepancies cause them to be considered no further in this reliability analysis. They neither add nor detract from the operational system in a critical sense and inclusion of this data would skew the significance of the reliability analysis. Another reason for eliminating this set of data from further analysis is their lack of timely repair. It was often considered more trouble than it is worth to repair these problems, and as the flight program progressed, even the reporting of these DRs were considered incomplete. The occurrence of a critically 1 or 2 DR would require some type of resolution before further flight, by way of a software repair, system work-around (function maintained by a backup), or it was carried as an accepted risk. Whatever the resolution, it was determined as highly improbable that the error would occur and cause system failure, and in reality none did. However, repair was not mandated for the criticality level 3 DRs and they were fixed at a convenient time.

Criticality level 2 DRs were written on problems that could compromise the research mission of the aircraft. The research program was designed to incrementally push the boundaries of the advanced technologies being demonstrated on the X-29, thus the flight envelope was continually revised throughout the program. The flight control system software was designed to support this expanding capability, however, the core, safety-critical functions remained unchanged. The criticality level 1 DRs thus represent the test experience of a stable software core, and it is this subset of reports that are studied in-depth, following a model selection process using both critically level 1 and 2 data. The analysis process and results are presented in the following sections.

## 6 STATISTICAL ANALYSIS USING SMERFS

To determine the applicability of existing software reliability models to critical real-time systems, the SMERFS program (Statistical Modeling and Estimation of Reliability Functions for Software) was executed on the X-29A data [4]. This package provides a representative selection of current software reliability models in a user-friendly form. SMERFS allows the user flexibility in the format of the input data. Of the four types of data allowed, two were selected for this analysis: *test interval* data with number of discrepancies reported within that time, and *CPU TBE* data, which presents the actual CPU execution time between discrepancies.

SMERFS provides five models for the *CPU TBE* type data and five models for the interval type data. Detailed information on the program's capabilities and the various available models can be found in [5, 6]. The formatted data were appropriate for execution of nine of these models. The NHPP model in the *CPU TBE* data type would not run, indicating that the "data are not appropriate for this model." The following models were executed: (a) *CPU TBE*—Littlewood/Verrall, Geometric, Musa Basic Execution, and Musa Logarithmic–Poisson Execution Time, and (b) test interval—Generalized Poisson,

Nonhomogeneous Poisson, Brooks & Motley Discrete, Schneidewind Maximum Likelihood, and S-Shaped Reliability Growth.

SMERFS was initially executed on the criticality 1 and 2 data, consisting of 37 total discrepancies. After selection of two models, further analysis is performed on the criticality 1 data only. There were a total of 12 criticality 1 discrepancies in approximately 2900 CPU hours of full up systems operations. The interval data was formatted by separating the total test hours into 100-hour intervals. The results obtained from these data sets are presented.

### 6.1 Model Assumptions

Each model operates on a set of assumptions and the user is to decide the applicability to the data. All models executed assume that the subject software was operated in a similar manner as anticipated for operational use. This was true for the X-29A criticality level 1 data since the safety-critical core software was considered a fixed entity. Only assumptions regarding the immediate repair of the software without new error introduction were generally applicable—no subsequent discrepancies reported were traceable to previous repairs. The system was maintained by those with intimate knowledge of the design and integration. If the software was not altered immediately after a discrepancy was noted, then the path in question was ensured not to be executed. No noted errors or discrepancies were traced back to the introduction of "code fixes."

### 6.2 Model Test Strategy

Attempts were made to execute 10 software reliability models for two data formats: *CPU TBE* (central processing units time-between-error) and *interval counts and lengths*. Nine of the 10 models executed properly. The models were executed on the criticality 1 and 2 data. Figures 7a and 7b show that there was no single model that obviously fit this data better than the others and that further analysis was required.

The remainder of the analysis is performed only on the criticality 1 data. This consisted of executing each model on this data set and comparing both the retrodictive fit (how well the model fit the input data) and the predictive fit (input $t - 1$ data points and have the model predict the next). One model from each data type was then selected for the final comparison on predictive accuracy.

### 6.3 *CPU TBE* Data and Model Executions

The data for the *CPU TBE* models are shown in Table 2. The DR number is associated with the time to that failure since the previous failure. The last column of Table 2 is the remainder of the test and flight time in which no discrepancies were reported. Execution of the four models available for this type of data was attempted in a stepwise manner to investigate the predictive capabilities of each of the models. That is, upon entering the $t_1$ to $t_{i-1}$ data values, the models were requested to predict the MTBF for $t_i$. The predictions requested of the models were for $t_6$ through $t_{13}$. The results are presented in Table 3. For

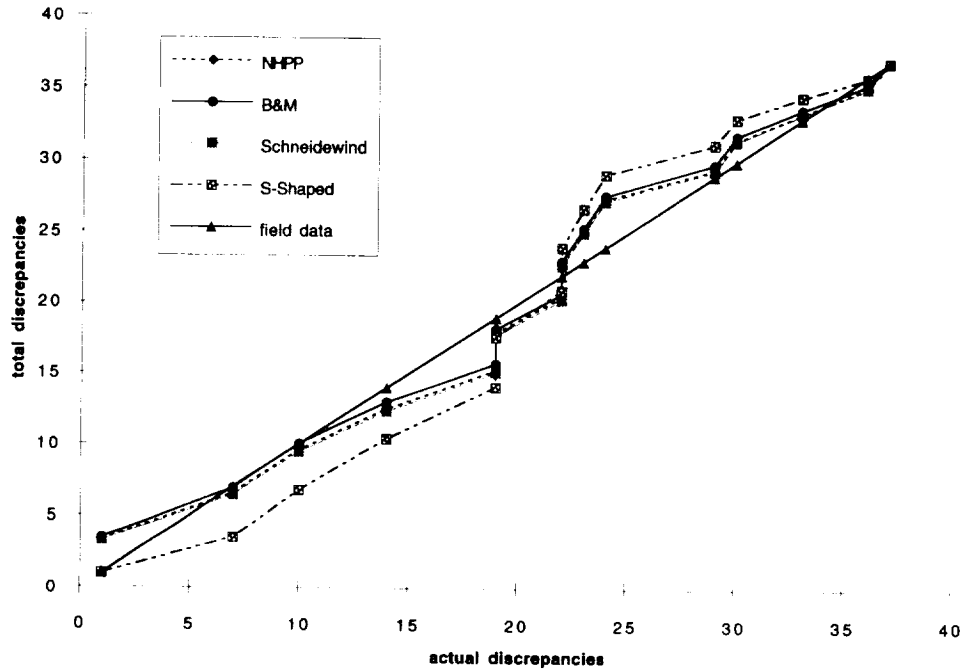


**Figure 7a. Criticality 1 and 2 test interval field data vs. retrodictive fit.**

example, Littlewood/Verrall's model executing the maximum likelihood estimate with linear fit operating on actual data $t_1$ through $t_9$ predicted the MTBNE for $t_{10}$ is 176.21 hours. As seen, not all the models could produce valid results on this set of data. The term **na** indicates that the model did not yield a prediction. This is because of the inability of the models to establish internal parameters within the confines of the programs trust region, primarily due to the rapid change in data.

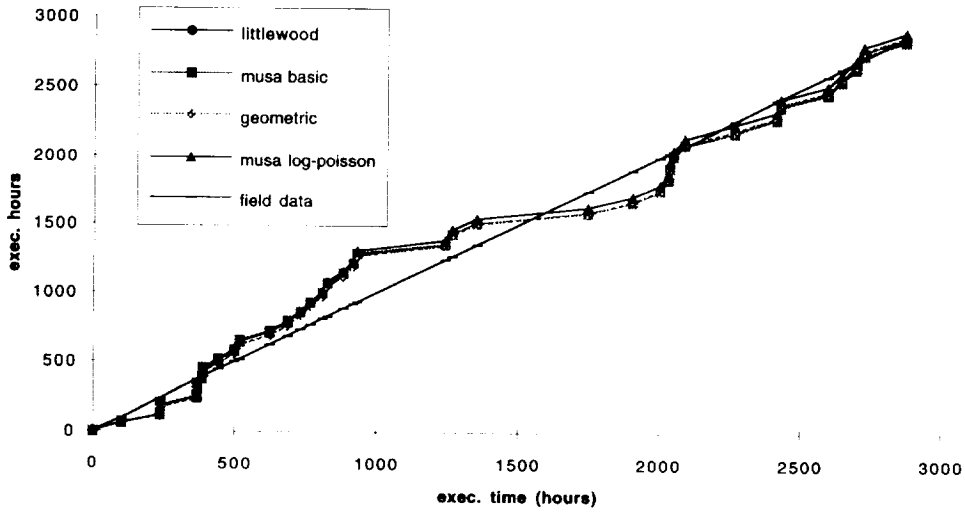


**Figure 7b. Criticality 1 and 2 *CPU TBE* field data vs. retrodictive fit.**

**Table 2.** ***CPU TBE*** **Data**

| dr # | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | addl. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *CPU TBE* | 237 | 1 | 130 | 0 | 20.5 | 53.5 | 322.5 | 45 | 15.5 | 106.5 | 1333 | 329.5 | >269 |

The results from the models varied widely. There is no obvious consensus on the predictions from one failure to the next. Of the computed predictions, the Geometric least squares is extremely optimistic. Failing to provide results for $t_{10}$ and $t_{11}$ causes no further analysis with this model. Musa's models were consistent in that no results were provided until $t_{12}$. Hence, this model was not reliable for providing stepwise predictions early in the flight program. The Littlewood/Verrall (LV) model consistently produced results, however. Between the four versions of this model, the maximum likelihood values tended to be optimistic in times $t_9$ and $t_{10}$ yet compensated for the late failures in times $t_{11}$ and $t_{12}$. Opposite from the maximum likelihood, the least squares version of the LV tended to react more quickly to the most recent data without overcompensating the future predictions. Hence, this model (LV least squares/quadratic fit) is selected to represent the *CPU TBE* data type models for further analysis. Both the retrodictive fit and the predictive fit provided by the LV model are shown in Figure 8, overlaid onto the actual data.

### 6.4 Test Interval Model Execution

For analysis on the test interval type models, the data in Table 2 were combined into 27 intervals, each representing 100 execution hours. Each available model for this data type was executed in both a step-wise predictive and the retrodictive manner. The predictions for the number of errors occurring in the next test interval (of 100 hours) from these type

**Table 3.** ***CPU TBE*** **Model Predictions For $t_i$ Given $t_1$ to $t_{i-1}$**

| Name → Technique → | LV ml/lin | LV ml/quad | LV ls/lin | LV ls/quad | Geom ml | Geom ls | MusaB ml | MusaLP ml |
|---|---|---|---|---|---|---|---|---|
| t6 | na | na | na | na | na | na | na | na |
| t7 | na | na | na | na | na | na | na | na |
| t8 | na | na | 145.2 | 211.5 | 130.79 | 1710.4 | na | na |
| t9 | 239.3 | 104.43 | 104.11 | 130.57 | 103.30 | 105.99 | na | na |
| t10 | 176.21 | 176.21 | 65.37 | 66.34 | na | na | na | na |
| t11 | 101.25 | 134.14 | 76.56 | 81.96 | na | na | na | na |
| t12 | 268.4 | 300.69 | 530.4 | 743.2 | 448.5 | 16270 | 962.8 | 225 |
| t13 | 12727 | 2775.1 | 517.55 | 678.49 | 497.2 | 922.8 | 798.75 | 239 |

*Note.* ml = maximum likelihood estimation; ls = least squares estimation; lin = linear fit; quad = quadratic fit.

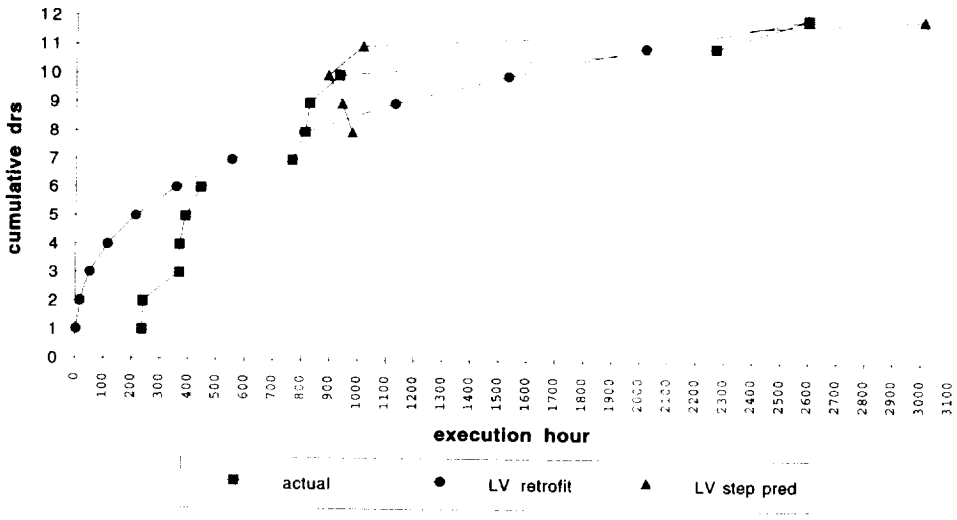


**Figure 8. Littlewood/Verrall cumulative discrepancies.**

models were not as informative as with the previous type. This is because the input range was from 0 to 3 discrepancies occurring in a testing period. The models did not output whole numbers, but rather fractions. However, SMERFS provides another prediction from these types of models (with the exception of Schneidewind) which is the estimated total number of errors at a given time. Many software reliability models calculate, based on the input, the total number of errors within the system in order to determine the model parameters and the tendency of the curve fit. The total number of errors predicted at various stages by these models is provided in Table 4. As can be seen, the reactiveness

**Table 4. Estimated Total Number of Errors**

| | GP ml | GP ls | NHPP ml | NHPP ls | BM Binom. | BM Poisson | S-Shape |
|---|---|---|---|---|---|---|---|
| t4 | 17376 | na | na | na | na | na | na |
| t8 | 10.3 | 13.7 | 67 | 101 | 9 | 8 | 9 |
| t9 | 70 | 77 | na | na | 125 | 57 | 16 |
| t10 | 97 | 104 | na | na | 72 | 81 | 16 |
| t12 | 13.5 | 17.1 | 21 | 28 | 11.4 | 11.5 | 11.6 |
| t17 | 10.0001 | 12.1 | 11 | 13.5 | 10 | 10 | 10.2 |
| t22 | 10.00 | 11.27 | 10.2 | 12.2 | 10 | 10 | 10 |
| t25 | 11.00 | 11.74 | 11.5 | 12.8 | 11 | 11 | 11 |
| t27 | 12.012 | 12.393 | 13.1 | 13.5 | 12 | 12 | 12.2 |

that was discovered in the *CPU TBE* type models is now shown in these models with this output.

All of the interval data models provided comparable fits to the data; however, in retrospect, the S-Shaped model consistently provided good predictions to the total number of errors. It did not seem to overreact to failure intense intervals, but rather remained consistent with its previous estimations and maintaining a real sense of limited total number of errors. Figure 9 shows the S-Shaped model retrodictive fit as well as the stepwise predictions, with the actual data. This model is selected to be the interval data representative model for further analysis in the next section.

### 6.5 Comparative Analysis of Predictive Capabilities

There are three primary aspects of a statistical reliability model output: the probabilistic model, a statistical inference procedure for determining the model parameters, and a prediction procedure that combines the previous two to provide probability statements about future failures [7]. Furthermore, it is not just the application of each model that yields reliability results, but the validation of the accuracy of the models that provides the credibility of the results. This validation comes only from analyzing each contributing aspect of the model to the predictions as well as their outputs. It is common for some models to provide accurate predictions in total number of errors but possibly be far off in time to next error. It is only in the full analysis of each aspect of the models that will indicate true validity. Several measures to validate all aspects of reliability models are documented [7], including the measurement of model noise and bias, median variability, rate variability, and prequential likelihood. Hence to deduce the predictive accuracy of models, several tests should be performed on the outputs. Although these methods are recognized as the proper means to truly understand how valid the models predict, the data

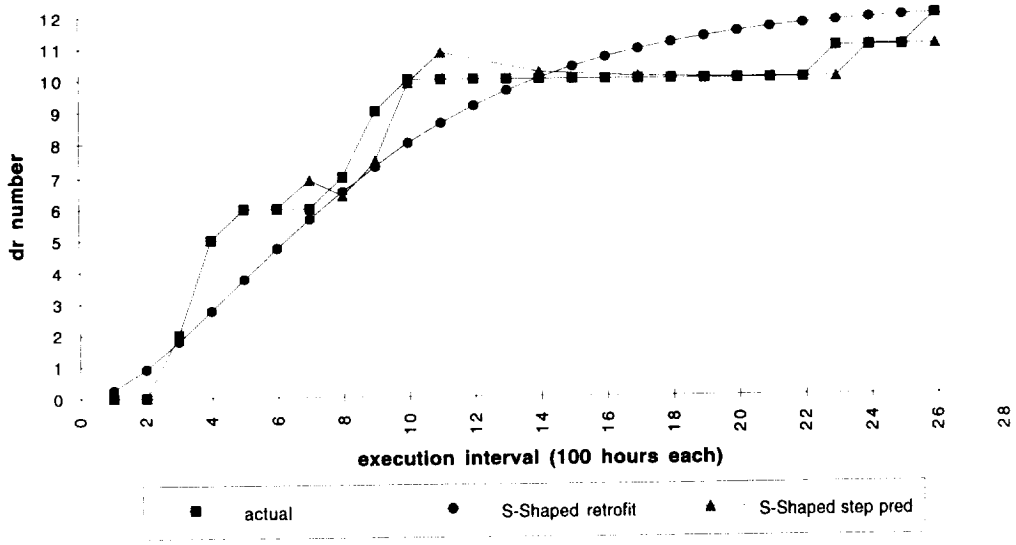


**Figure 9. S-Shaped cumulative discrepancies.**

provided by this phase of the X-29A do not provide enough individual results to provide a sufficient sampling to warrant these analysis functions. Furthermore, the methods proposed by Abdel-Ghaly et al. are to determine, for a given data set, which model(s) provide the best predictions. The intent of this paper is to determine if these models in general are applicable to this type of data set. The predictive accuracy analysis within the context of this paper is not to pick a *best model,* but rather to aid in the determination of their *general applicability.* Therefore, less sophisticated methods are used for this analysis [8]. Subsequent studies on safety-critical systems, provided more data, will utilize more comprehensive techniques, such as those proposed by Abdel-Ghaly et al.

To assess the applicability of the models for a predictive analysis on this data, the accuracy of the model fits and their capability to predict failures are analyzed. Both the Littlewood/Verrall least squares parameter estimation with quadratic fit from the *CPU TBE* model group and the S-Shaped from the interval-data models were selected for further analysis. The values obtained from these models, both stepwise predictions and the retrodictive model fit, are shown in Figures 8 and 9.

The results are compared using two relative error calculations—one for the cumulative number of discrepancies to a time $t_f$, called $\varepsilon 1$, and the other for the estimated interval time (MTTF) between failures $f_i$ and $f_{i+1}$, called $\varepsilon 2$ [8]:

$$\varepsilon 1 = |m(t_f) - \text{actual number}|/\text{actual number}$$

$$\varepsilon 2 = |1/\lambda(t_f) - \text{estimated value}|/\text{estimated value}$$

where $m(t_f)$ is the expected number of cumulative failures at time $t_f$ and $\lambda(t_f)$ is the failure intensity at time $t_f$.

The $\varepsilon 1$ values were calculated using the cumulative number of failures at respective times, as presented in Table 5. Because the data set is sparse, the estimated cumulative number of failures used the sole estimated values as provided by the models—not averages. The values are presented as rounded to the nearest whole number. The LV could not provide predictions by $t = 500$. Based on the relative error plot as shown in Figure 10, it is determined that the S-Shaped prediction set provided the overall best estimate of cumulative number of errors at these given times. The LV model, however, provided a better retrodictive fit.

**Table 5. Cumulative Failures For Time *t***

| Time *t* | LV Pred | LV Retro | S-Shape Pred | S-Shape Retro | Actual |
|---|---|---|---|---|---|
| 500 | na | 6 | 6 | 4 | 6 |
| 1000 | 10 | 8 | 10 | 8 | 10 |
| 1500 | 11 | 9 | 10 | 10 | 10 |
| 2000 | 11 | 10 | 10 | 11 | 10 |
| 2500 | 11 | 11 | 11 | 12 | 11 |
| 2700 | 11 | 12 | 11 | 12 | 12 |

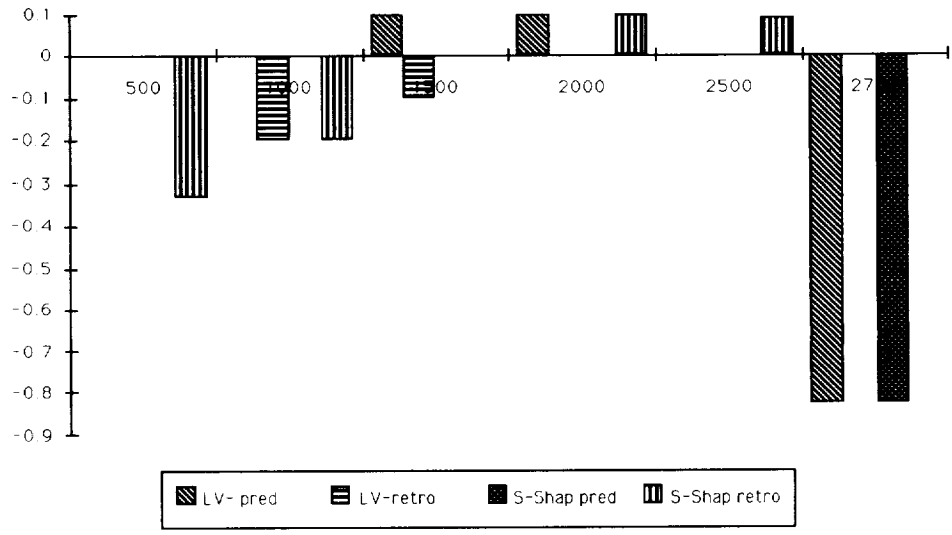


**Figure 10. Relative error for cumulative number of failures. ε1.**

The relative error for the mean time to failure is provided through calculations of ε2. The graphs in Figures 6 and 7 indicate the models' predictions for these times, both stepwise and retrodictive. Because the S-Shaped model operated on interval data and produced an estimated number of failures for the next interval, it did not yield the actual time to next failure. This time was computed by interpolating on the graph between the data points to assess the estimated time between failures. As in the estimates for use of ε1, the estimated time to failure used here are not averages but are the sole estimated values by these models. The data for each of the models are provided in Table 6 with their respective ε2 values in Figure 11. As shown, the predictions from the models were not available until the eighth discrepancy, however the ε2 is calculated for discrepancy 6–12.

As shown in Table 6, the difference between the MTTF of the models and the actual are

**Table 6. Estimated Time to Next Failure**

| Failure Num | Actual TTF | LV Pred | LV Retro | S-Shape Pred | S-Shape Retro |
|---|---|---|---|---|---|
| 6 | 53.5 | na | 143 | na | 110 |
| 7 | 322.5 | na | 195.4 | 237.5 | 120 |
| 8 | 45 | 211.5 | 255.8 | 251 | 140 |
| 9 | 15.5 | 130.57 | 324.2 | 112 | 170 |
| 10 | 106.5 | 66.34 | 400.7 | bin w/f9 | 230 |
| 11 | 1333 | 81.96 | 485.2 | infinity | 330 |
| 12 | 329.5 | 743.2 | 577.8 | infinity | 880 |

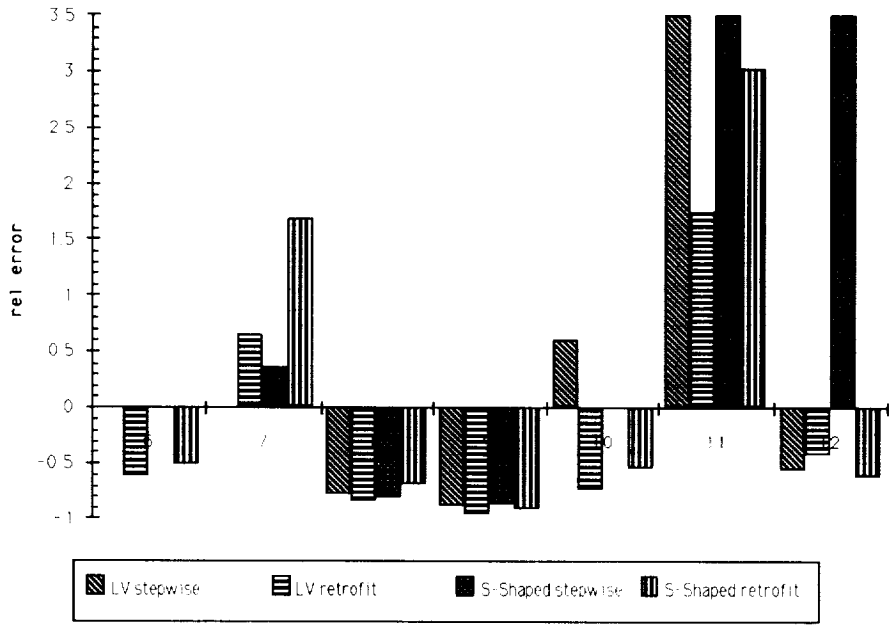


**Figure 11. Relative error for MTTF, ε2.**

not surprising. The S-Shaped prediction for failure 10 is not provided as it occurred in the same time interval as failure 9. For this reason, the S-Shaped model did not predict for that failure, but rather for that interval it had predicted 1 failure to occur. Given the small data set and the wide range in values from one discrepancy to the next, this data seem difficult to fit and more difficult to predict. It is interesting to see, though, the difference in the same model from the predictive stage to the retrodictive stage. The LV model had the largest gap between predictive and retrodictive for the 10th and 11th discrepancies. The S-Shaped predictive tended to become optimistic with the increased number of nonfailed intervals, indicated by the model estimating infinite time to next failure (all of the failures had been found). The predicted time to the last two failures was infinity. The retrodictive fit of the S-Shaped model seemed to fit the failures better.

The relative error plot of this MTTF data (ε2), provided in Figure 11, shows an error consistent among the models up through failure number 10. The stepwise predictions did not fare well on the estimates for numbers 11 and 12; however, given the small set of data and its wide range of values, one is not surprised. Based on these values, the tendency is to be turned away from the S-Shaped predictive because of the last two values being off the chart, and more toward the LV retrofit, which remained on the chart with consistently less error than the S-Shaped retrofit. This disproved the previous intuitive selection of the S-Shaped model, which was based on how the model seemed to fit.

## 7 OBSERVATIONS

The flight test community at Dryden Flight Research Facility has established a credible means of testing and achieving reliability for safety-critical flight systems. To date, their

operations have not included the use of formal software reliability modeling techniques. The success of the many programs at Dryden attests to the value of their methods, with this independent evaluation of the X-29A flight control software indicating no difference. However, the X-29 flight test program was one of Dryden's most successful programs. Acknowledging that not all programs are carried out as smoothly as this, the use of objective analysis tools could provide keen insight to problem areas. The analysis of the X-29A flight data did not indicate that the program was unreliable, and it did not dispute the achieved results. Thus the internal engineering judgment and post facto statistical model results were similar.

The results obtained through the statistical analysis using the stepwise prediction and retrodictive fit of several standard software reliability models were not strong enough to justify the use of any one of the models. More important, the predictive capabilities of the models were inhibited by the sparseness and variability of the criticality level 1 data. Some observations made in this analysis are:

- Various levels of criticality and their implications of failure suggest that a visibility factor affects error detection. This is opposite from assumptions underlying some of the models that either all errors have the same probability of being detected or are all of the same criticality.
- The data set used within this analysis was sparse and difficult for the models to fit. However, this data set was complete. The models were not usable until half of the failures were discovered as it required between 5 months and 2 years of full-up systems testing data (5–10 data points) to produce output. In this safety-critical application, if the manager were awaiting reliability results, the flight program would have been unduly delayed.
- The use of software reliability analysis techniques can present useful information in an operational context. These techniques are to aid the analysis of current status and provide predictions for future error occurrences. With continual updates on the "raw data" in the form of graphs and tables, intuitive analyses can aid the program managers in deciding to test further or to possibly even discontinue use of a current version of code and proceed to the next.
- The models do not utilize all of the data available for assessing and predicting the reliability. Perhaps there are indicators that could be extracted from other relevant project information that would enable results sooner—especially given that the data are sparse.
- The statistical models provided by the software reliability community are designed to provide probabilistic predictions for time and occurrence of future discrepancies. This type of information can also support the safety-critical systems operational community. However, it is only through the publication and sharing of safety-critical systems failure data like the X-29A that the reliability modeling community can adjust their techniques and models to more accurately represent similar safety-critical system behavior.
- After establishing the applicability of statistical reliability models, evaluation techniques for the validity of the predictions must be applied to every contributing aspect of the models. The techniques utilized within this paper were necessarily crude due to the small data set, however use of referenced methods are preferred for completeness.
- Even if statistical results are not heavily relied upon, the heightened awareness created

through the establishment and use of various data collection mechanisms with at least quick look analyses would realize benefits in reliability for any program.

More information from the program could and should be incorporated into the models to enable more rapid production of quality results. Finer tuned models designed for this type of operational software have the potential to enable prediction of error occurrence and focus software testing for specific errors by criticality level. One of the greatest factors in reliability analysis of real-time, safety-critical systems is the implications of failure. With human life at risk, a purely statistical representation of the assessed reliability is not enough on which to base a life-critical decision to fly, but this information may be useful in determining when NOT to fly.

## 8 CONCLUSIONS

Flight programs have evolved in complexity while at the same time flight schedules have become more aggressive. Traditionally, software reliability models have been developed for and on commercial software systems, where the user base is large with exposure to a wide variety of operational scenarios. However, for many safety-critical systems, especially within the research community, the target software is typically not duplicated beyond several instances and the operational testing is within the local community (as opposed to hundreds of users testing desktop application software). The implications of failure are also much greater within a safety-critical system. Many engineers are inclined to believe that their software is naturally robust, and the present failure is the proverbial "last bug." These are often reasons why failure data are not collected and maintained, as it is oftentimes the notion that each failure is an anomaly unto itself and that there is no correlation among subsequent failures. This is not intuitively true and only a complete set of collected failure data would substantiate or negate this idea.

Software is increasingly used in safety-critical systems, with its design and functionality becoming more complex. Tools and methods must be developed for use within the operational community that enable objective assessment of the reliability of the software system. One perspective that would provide aid for software reliability assessment is spawned from the hardware community and is based on occurrence of errors with respect to CPU time. The raw data graphs themselves provide the managers with an intuitive look at whether the code is becoming more or less reliable and hence help them to decide whether or not to fly. The use of statistical models would add credence to this judgment and ideally provide predictive capability on time of next error and total number of errors. These predictions can be validated by use of reported evaluation techniques.

A systems engineering approach to validating the flight control system entails utilizing full-up systems testing, operating the system in a manner in which it will be executed in actual flight, and running through the operational scenarios in a rigorous yet timely manner. This method has proved reliable to the Dryden community in the case of the X-29A flight control program. However, the incorporation of reliability assessment methods within the context of any program, including X-29 program, could enable better resource scheduling and utilization, provide insight into the intensity as well as density of reported failures, and provide test engineers and managers with objective measurements within a subjective program.

The use of software reliability analysis techniques is achievable in any system involv-

ing software. Independent of the source language or the underlying philosophy (modular, structured, AI, neural networks), the analysis is the same because the system is evaluated objectively from a "black-box" perspective. It is the results of the modeling and prediction that allow for subjective explanations. By exercising engineering judgment when viewing the information represented by the failure and CPU data, reliability assessment can be performed from both subjective and objective views.

## ACKNOWLEDGMENTS

The authors would like to thank the following people for their helpful comments during the course of this study and on the paper: Bill Farr, designer of SMERFS; Ed Collins, X-29 program historian; and the reviewers, for their invaluable comments on the analysis techniques and presentation of the results. Many thanks also to Peter Cheeseman, Mark Boyd, and Lee Duke for their comments. And thanks to Chris Flores and his staff at Dryden for being cooperative and supportive, and for providing us with the data in an organized and timely manner.

## REFERENCES

[1] M.R. Earls and J.R. Sitz, "Initial flight qualification and operational maintenance of X-29A flight software," NASA Technical Memorandum 101703, 1989.

[2] E.L. Duke, "Software V & V of flight and mission-critical software," *IEEE Software*, May 1989.

[3] M.-H. Chen, A.P. Mathur, and V.J. Rego, "Effect of testing techniques on software reliability estimates obtained using time-domain models," in *IEEE Proc. Software Reliability Symposium*, Denver, CO, 1992.

[4] W.H. Farr, "Statistical modeling and estimation of reliability functions for software (SMERFS)," Naval Surface Warfare Center, NAVSWC TR 84-373, Rev. 2, Mar. 1991.

[5] W.H. Farr, "A survey of software reliability modeling and estimation," Naval Surface Weapons Center, NSWC TR 82-171, Sep. 1983.

[6] W.H. Farr, "Statistical modeling and estimation of reliability functions for software (SMERFS) library access guide," Naval Surface Warfare Center, NAVSWC TR 84-371, Rev. 2, Mar. 1991.

[7] A.A. Abdel-Ghaly, P.Y. Chan, and B. Littlewood, "Evaluation of competing software reliability predictions," *IEEE Trans. Software Engineering*, vol. SE-12, no. 9, Sep. 1986.

[8] F.M. Vallee and A. Ragot, "Reliability evaluation using NHPP models," in *IEEE Proc. International Symposium on Software Reliability Engineering*, Austin, TX, 1991.

**Gloria J. Davis** received a B.S. in Computer Science from the University of New Mexico in 1983 and an M.S. in Computer Science in 1986 from Santa Clara University. In 1980 she began working at NASA Ames Research Center with the Flight Systems and Simulation Research Division. Her focus on fault-tolerant flight control systems continued through 1987. In 1988 she transferred to the Information Sciences Division. She is currently a member of the Dependable Systems Research Group and focuses primarily on software reliability analysis research. She is also a member of the NRC Panel on Statistical Methods in Software Engineering.

**Michael R. Earls** received a B.S. in Electronic Engineering from California State Polytechnic University at Pomona in 1966. He began working at NASA's Dryden Flight Research Facility in the Guidance & Displays Branch as a hardware engineer on the X-15 alpha-beta sensor (ball nose). Upon completion of that program, he worked on the navigation system of the YF-12 aircraft. The HiMAT program, a supersonic remotely piloted vehicle, was his initiation to real-time system software, where he wrote and tested the onboard control system software. He then joined the X-29 program (before first flight of ship 1) and designed the hardware for the simulator actuator models. He became the software manager and remains such for the X-29 ship 2. He is also currently associated with the software of the F-18 HARV research flight control system.

**F.A. Patterson-Hine** received a B.S. in Mechanical Engineering from the university of Alabama in 1981, an M.S. in 1984, and a Ph.D. in 1988, both from the University of Texas at Austin in Mechanical Engineering. She was a NASA Graduate Student Research Fellow and a ZONTA Amelia Earhart Fellow. She is currently the Dependable Multiprocessing Group Leader in the Information Sciences Division at NASA Ames Research Center. Her interests include modeling and analysis of complex systems, model-based fault diagnosis, fault management in real-time systems, and software reliability measurement and modeling. She is a member of Tau Beta Pi, Phi Kappa Phi, the IEEE Computer Society and Reliability Society, the Association for Computing Machinery, and the American Nuclear Society. She is a Registered Professional Engineer in California.